CLAIMS

1. (Original) A method for estimating a query compilation time of a query optimizer comprising the steps of:

(a) receiving a query;

(b) iterating through possible join pairs for said query;

(c) for each join pair, identifying a set of differentiating properties and using said identified set of differentiating properties to calculate number of join plans; and

(d) estimating the compilation time from said calculated number of join plans for each type of join method.

2. (Original) A method as per claim 1, wherein said join pairs are iterated by reusing existing join enumerator in said query optimizer.

3. (Original) A method as per claim 1, wherein plan generation in said query optimizer is bypassed.

4. (Original) A method as per claim 1, wherein said set of differentiating properties comprises any of, or a combination of the following: order, data partition, pipelineability, data source, and presence of expensive predicates.

5. (Original) A method as per claim 1, wherein said query is an SQL query.

6. (Original) A method as per claim 1, wherein said estimation is performed via a regression model.

7. (Original) A method as per claim 1, wherein said compilation time is estimated via running regression of the following model:

$$T = T_{inst} \times \sum (C_t \times P_t)$$

wherein T is a machine-dependent parameter representing time per instruction, C_t is a constant representing number of instructions to generate a join plan of type *t*, and P_t is an estimated number of join plans of type *t* as estimated in step (d).

8. (Original) A method as per claim 1, wherein said differentiating properties are generated in any of the following policies: a lazy policy in which said differentiating properties are generated naturally or an eager policy in which said differentiating properties are forcibly generated via an optimizer.

9. (Original) A method as per claim 1, wherein said number of join plans are calculated for any join type selected from a group consisting of: nested loops join (NLJN), sort merge join (MGJN), and hash join (HSJN).

10. (Original) A method as per claim 1, wherein compilation time for multiple optimization levels are estimated in a single pass.

11. (Original) A compilation time estimator (COTE) bypassing plan generation in a query optimizer and reusing a join enumerator to estimate compilation time of said query optimizer, said join enumerator iterating through possible join pairs for a query, and, for each join pair, said COTE identifying a set of differentiating properties and using said identified set of differentiating properties to calculate number of join plans, and estimating compilation time from said calculated number of join plans for each type of join method via a regression model as follows,

$$T = T_{inst} \times \sum (C_t \times P_t)$$

wherein T is a machine-dependent parameter representing time per instruction, C_t is a constant representing number of instructions to generate a join plan of type *t*, and P_t is an estimated number of join plans of type *t*.

12. (Original) A compilation time estimator (COTE), as per claim 11, wherein said set of differentiating properties comprises any of, or a combination of the following: order, data partition, pipelineability, data source, and presence of expensive predicates.

13. (Original) A compilation time estimator (COTE), as per claim 11, wherein said differentiating properties are generated in any of the following policies: a lazy policy in which said differentiating properties are generated naturally or an eager policy in which said differentiating properties are forcibly generated via a optimizer.

14. (Original) A compilation time estimator (COTE), as per claim 11, wherein said number of join plans are calculated for any join type selected from a group consisting of: nested loops join (NLJN), sort merge join (MGJN), and hash join (HSJN).

15. (Original) A compilation time estimator (COTE), as per claim 11, wherein said query is an SQL query.

16. (Original) A system for estimating query compilation time via reusing a join enumerator in a query optimizer, said system comprising:

(a) an interface to receive a query;

(b) a join enumerator to iterate through possible join pairs for said query, said iteration performed via reusing said join enumerator in said query optimizer;

(c) a property identifier to identify, for each join pair, a set of differentiating properties and use said identified set of differentiating properties to calculate number of join plans; and

(d) a compilation time estimator to estimate compilation time from said calculated number of join plans for each type of join method, wherein said number of join plans are calculated for any join type selected from a group consisting of: nested loops, sort merge, and hash.

17. (Original) A system as per claim 16, wherein said set of differentiating properties comprises any of, or a combination of the following: order, data partition, pipelineability, data source, and presence of expensive predicates.

18. (Original) A system as per claim 16, wherein said compilation time estimator uses a regression model to estimate said compilation time.

19. (Original) A system as per claim 16, wherein said compilation time is estimated via running regression of the following model:

$$T = T_{inst} \times \sum (C_t \times P_t)$$

wherein T is a machine-dependent parameter representing time per instruction, C_t is a constant representing number of instructions to generate a join plan of type *t*, and P_t is an estimated number of join plans of type *t* as estimated in step (d).

20. (Original) A system as per claim 16, wherein said differentiating properties are generated in any of the following policies: a lazy policy in which said differentiating properties are generated naturally or an eager policy in which said differentiating properties are forcibly generated via a optimizer.

21. (Original) A system as per claim 16, wherein said query is an SQL query.

22. (Original) An article of manufacture comprising computer usable medium having computer readable program code embodied therein estimating a query compilation time of a query optimizer via reusing an existing join enumerator in said query optimizer, said medium comprising:

(a) computer readable program code aiding in receiving a query;

(b) computer readable program code iterating through possible join pairs for said query;

(c) for each join sequence, computer readable program code identifying a set of differentiating properties and using said identified set of differentiating properties to calculate number of join plans; and

(d) computer readable program code estimating compilation time from said calculated number of join plans for each type of join method.

23. (Original) An article of manufacture as per claim 22, wherein said estimation of compilation time is performed via a regression model.

24. (Original) An article of manufacture as per claim 22, wherein said set of differentiating properties comprises any of, or a combination of the following: order, data partition, pipelineability, data source, and presence of expensive predicates.

25. (Original) An article of manufacture as per claim 22, wherein said compilation time is estimated via running regression of the following model:

$$T = T_{inst} \times \sum (C_t \times P_t)$$

wherein T is a machine-dependent parameter representing time per instruction, C_t is a constant representing number of instructions to generate a join plan of type *t*, and P_t is an estimated number of join plans of type *t* as estimated in step (d).

26. (Original) An article of manufacture as per claim 22, wherein said differentiating properties are generated in any of the following policies: a lazy policy in which said differentiating properties are generated naturally or a eager policy in which said differentiating properties are forcibly generated via a optimizer.

27. (Original) An article of manufacture as per claim 22, wherein said number of join plans are calculated for any join type selected from a group consisting of: nested loops join (NLJN), sort merge join (MGJN), and hash join (HSJN).

28. (Original) A method for estimating query compilation time in a query optimizer, said method comprising the steps of:

bypassing plan generation and reusing a join enumerator of said query optimizer to identify number of joins;

iterating through possible pairs for a query;

for each join, accumulating a set of differentiating properties during enumeration and using said identified set of differentiating properties to calculate number of join plans; and

estimating compilation time from said calculated number of join plans for each type of join method via a regression model.

29. (Original) A method as per claim 28, wherein said set of differentiating properties comprises any of, or a combination of the following: order, data partition, pipelineability, data source, and presence of expensive predicates.

30. (Original) A method as per claim 28, wherein said query is an SQL query.

31. (Original) A method as per claim 28, wherein said compilation time is estimated via running regression of the following model:

$$T = T_{inst} \times \sum (C_t \times P_t)$$

wherein T is a machine-dependent parameter representing time per instruction, C_t is a constant representing number of instructions to generate a join plan of type *t*, and P_t is an estimated number of join plans of type *t* as estimated in step (d).

32. (Original) A method as per claim 28, wherein said differentiating properties are generated in any of the following policies: a lazy policy in which said differentiating properties are generated naturally or an eager policy in which said differentiating properties are forcibly generated via a optimizer.

33. (Original) A method as per claim 28, wherein said number of join plans are calculated for any join type selected from a group consisting of: nested loops join (NLJN), sort merge join (MGJN), and hash join (HSJN).